**ITS Data Warehouse**

**Project Reference Document**

**Version 1.0**

**Author: Madhubabu Sandara**

**MS in Computer Science**

**SUNY Buffalo**

**Date: 01/22/2010**

**Technologies Used to Implement the Project:**

JDK 1.6

Spring Framework,

Hibernate Object Relational Mapping (ORM) API,

Quartz Scheduler API for Batch Programs (For Data Import of XML Files)

Apache ANT

*XML Processing APIs:*

JAXB,

StAX Parser,

*Others:*

AJAX,

HTML,

Cascading Style Sheets (CSS)

**IDE Used for Source Code Development:**

Eclipse IDE – Java/J2EE Helios (Preferable)

**Web Server:**

Apache Tomcat 6.0.29

Applications running on the Web Server

1. GeoServer (geoserver.war)
2. ITS Web Application (ITSWebApp.war)

**Database Used:**

MySQL

**‘WAR’ means “Web Archive”**

**Project Structure:**

**/ITSWebApp** (Root Directory)

**/build.properties 🡪**

Build Properties file – Tomcat deployment directory, Username and Password to connect to Tomcat Web Server for Automatic deployment

**/build.xml 🡪**

Apache Ant build file for compiling and deploying source code to Tomcat Web Server.

**src** 🡪 java classes

**data/xml** 🡪 XML files

**data/xsd** 🡪XML Schema files

**tools\_and\_libs** 🡪 Third Party Tools and Libraries Used

**WebContent/css** 🡪 CSS files for User Interface (may include 3rd party files)

**WebContent/images** 🡪 images used in the application

**WebContent/map\_files** 🡪 Map files which includes CSS and Images

**META-INF** 🡪 Never Mind!!!

**scripts/dhtmlx** 🡪 3rd party DHTMLX API for Front-end User Interface (UI) for displaying results in Grid.

**scripts/zapatec** 🡪 3rd party ZAPATEC API for Front-end User Interface (UI) for displaying Calendar Controls for selecting Data and Time.

**WEB-INF/classes 🡪** Compiled Java Class files.

**WEB-INF/jsp 🡪** Java Server Pages (JSP) files

**WEB-INF/lib 🡪** Library files (\*.jar files) used to compile Java source code (includes Spring, Hibernate etc.)

**WEB-INF/classes/batchjob.properties 🡪** Absolute Path to XML Files to read from.

**WEB-INF/classes/jdbc.properties 🡪** Not Used!!!

**WEB-INF/classes/log4j.properties 🡪** Log4J API properties file

**WEB-INF/applicationContext.xml 🡪**

Details about

Spring Java Bean Classes and their dependencies,

Quartz Scheduler Batch Program Configuration and Scheduling details,

MySQL Database configuration (Database URI, Username and Password)

Hibernate Mapping Files

**WEB-INF/ITSWebApp-servlet.xml** 🡪 Spring Mapping File

**WEB-INF/log4j.xml** 🡪LOG4J Configuration

**WEB-INF/web.xml 🡪** Web Application Deployment File

ITS Web Application is divided into THREE main categories

1. Database Schema
2. Data Import using Scheduled Batch Programs
3. Web Application – For designing Front-end User Interface and Querying MySQL Database.

**Database Schema:**

Tables:

LINK\_INFO

INCIDENTLOG

HELP\_LOG

DIRECTION

BCD\_EVENT\_HISTORY

DEVICE\_HISTORY

EXTERNAL\_AGENT\_HISTORY

TRAFFIC\_EVENT\_HISTORY

TRAVEL\_ROUTE\_HISTORY

*(Refer to MySQL DB SQL Scripts for the list of Columns in each table)*

**Data Import of XML Files:**

Use JAXB API to generate XML Binding files for Java Classes

**Reference Commands:**

**To generate XSD file from XML file – Use Trang library present in /ITSWebApp/tools\_and\_libs directory.**

Change directory to tools\_and\_libs/trang

At command line:

$ java –jar trang.jar sampleXmlFile.xml sampleXmlFile.xsd

This will generate the .xsd file from the sample xml file. This .xsd file is needed to generate bindings between xml file and java classes which we generate later.

*Refer to directory for current XML Binding Java Classes*

**src/edu/buffalo/csse/its/xmlbinding/\*.java files**

**Batch Programs**

*Refer to directory* - **src/edu/buffalo/csse/its/batchjobs/\*.java**

For each Batch Program, there are TWO Java files

Example: For Incident Log Batch Program to run, there are 2 files.

1. IncidentLogJob.java
2. IncidentLogTask.java

**/ITSWebApp/WEB-INF/applicationContext.xml** file has the Spring configuration and details about when to schedule these batch programs to run based on the CRON expression.

To Schedule Batch Programs using the CRON Expression in applicationContext.xml, please refer to <http://en.wikipedia.org/wiki/CRON_expression>

**/ITSWebApp/WEB-INF/classes/batch.properties** files has the location of XML files.

Make sure the directory containing XML files has both **READ** and **WRITE** permissions enables.

Each batch program reads the XML files at the location and processes it. Once a file is processed successfully and the data is imported to the database, the file is moved to a **\_done** directory.

**Steps to Deploy GeoServer to Tomcat:**

1. Download of use the existing GeoServer.war file. To download the geoserver-<version>-war.zip file, go to [www.geoserver.org](http://www.geoserver.org) website and download it. Unzip the file which contains the Web Archive (geoserver.war) file.
2. Copy the geoserver.war file to TOMCAT webapps directory
3. Go to TOMCAT\_HOME/bin directory, and run startup.bat to start the webserver. This will deploy the geoserver web application on Tomcat. If successful, you should be able to see “geoserver” dir under “webapps”dir.
4. Copy the Link Shapefiles to GEOSERVER web application dir.
5. Run GeoServer application and configure Link Groups by adding Links.

**Using ANT Tool to Deploy ITS Web App**

1. Run Eclipse IDE or one can use ANT command line to run commands. Using Eclipse is recommended for ease of use.
2. In Eclipse, select Window 🡪 Show View 🡪 Ant. This will add ANT to the current Eclipse perspective.
3. **Apache ANT tool should be installed and configured to use ANT in Eclipse. Set the Environment variables like ANT\_HOME etc. Please refer to Apache ANT Webpage for installation details.**
4. In Ant Window, right-click and add ‘build.xml’ file present in /ITSWebApp project directory.
5. You should be able to see different Targets present in the current ANT build file.
6. Make sure you configured ‘build.properties’ file before running any ANT targets.
7. After making any source code changes, use ANT targets like clean, build, deploy etc.
8. Select ‘clean’ then ‘build’ and ‘deploy’/’deploywar’ targets to deploy the source code changes to the TOMCAT/webapps directory directly. No need to manually copy the files to TOMCAT/webapps/ITSWebApp dir.
9. Restart TOMCAT Web Server to see the changes.
10. Open Web Browser and run <http://localhost:8080/ITSWebApp>
11. If everything runs without any errors, one should be able to see the application running which means the Batch Programs and the Web application should show up on the browser.
12. Refer to TOMCAT/logs for any possible errors when deploying the application. The information will be useful to debug possible errors or exceptions in the application.